

Metin Temizleme ve N-Gram Analizi

Öğrenci: Erkebulan Duishenaliev

Ders: BIL-479 Dil İşlemcileri

Öğretim Görevlisi: Abdülkadir Şeker

Tarih: Kasım 2025

Proje Özeti

Bu projede Türkçe metin üzerinde:

- Metin temizleme (preprocessing)
- Durak kelime çıkarma (stopword removal)
- Kök bulma (lemmatization)
- Metin üzerinde kapsamlı temizleme işlemleri:
- Küçük harfe çevirme (lowercase)
- Noktalama işaretlerini çıkarma
- Sayıları çıkarma (remove numbers)
- Fazla boşlukları düzenleme
- Tokenization (kelimelere ayırma)
- N-gram analizi (unigram, bigram, trigram) işlemleri gerçekleştirilmiştir.

GitHub Repository: <https://github.com/Erkebulan100/hw1/blob/main/hw1.ipynb>

1. Kütüphaneler

```
In [1]: import re
import string
from collections import Counter
```

2. Analiz için örnek metin

```
In [2]: text = """Türkiye'nin başkenti Ankara'dır. İstanbul en kalabalık şehirdir. Doğal gü
```

3. Gelişmiş Ön İşleme (Advanced Preprocessing)

```
In [3]: def advanced_clean_text(text):
        """Comprehensive text preprocessing"""
        # Convert to Lowercase
        text = text.lower()
```

```

# Remove punctuation
text = text.translate(str.maketrans('', '', string.punctuation))

# Remove numbers
text = re.sub(r'\d+', '', text)

# Remove extra whitespace
text = ' '.join(text.split())

return text

def tokenize(text):
    """Tokenize text into words"""
    return text.split()

```

```

In [4]: # Apply advanced cleaning
cleaned_advanced = advanced_clean_text(text)
print("After cleaning (with number removal):", cleaned_advanced)

# Tokenize
tokens = tokenize(cleaned_advanced)
print("\nTokens:", tokens[:10])

```

After cleaning (with number removal): türkiyenin başkenti ankaradır istanbul en kala balık şehirdir doğal güzellikler çok fazladır türk mutfağı dünyaca ünlüdür tarih ve kültür açısından zengin bir ülkedir

Tokens: ['türkiyenin', 'başkenti', 'ankaradır', 'istanbul', 'en', 'kalabalık', 'şehirdir', 'doğal', 'güzellikler', 'çok']

4. Stopword Removal

Anlamsız kelimeleri çıkarma.

```

In [5]: # Turkish stopwords (common words to remove)
turkish_stopwords = [
    've', 'bir', 'bu', 'da', 'de', 'den', 'için', 'ile', 'mi',
    'mu', 'mü', 'var', 'yok', 'gibi', 'daha', 'en', 'çok',
    'her', 'ne', 'o', 'şu', 'ya', 'ama', 'veya', 'ki', 'nasıl',
    'neden', 'niçin', 'bu', 'şu', 'o', 'ben', 'sen', 'biz', 'siz',
    'onlar', 'olan', 'olarak', 'kadar'
]

def remove_stopwords(text, stopwords):
    words = text.split()
    filtered_words = [word for word in words if word not in stopwords]
    return ' '.join(filtered_words)

```

```

In [6]: cleaned_no_stop = remove_stopwords(cleaned_advanced, turkish_stopwords)
print("\nText without stopwords:", cleaned_no_stop)

```

Text without stopwords: türkiyenin başkenti ankaradır istanbul kalabalık şehirdir doğal güzellikler fazladır türk mutfağı dünyaca ünlüdür tarih kültür açısından zengin ülkedir

5. Lemmatization (Kök Bulma)

Kelimeleri köklerine indirgemek için basit Türkçe stemmer. Yaygın ekleri (lar, ler, dan, den vb.) kaldırır.

```
In [7]: # Simple Turkish stemmer (basic suffix removal)
def simple_turkish_stemmer(word):
    """Remove common Turkish suffixes"""
    suffixes = ['lar', 'ler', 'dan', 'den', 'tan', 'ten', 'ın', 'in', 'un', 'ün',
                'nın', 'nin', 'nun', 'nün', 'ı', 'i', 'u', 'ü', 'da', 'de', 'ta', '']

    for suffix in suffixes:
        if word.endswith(suffix) and len(word) > len(suffix) + 2:
            return word[:-len(suffix)]
    return word

def lemmatize_text(text):
    words = text.split()
    lemmatized = [simple_turkish_stemmer(word) for word in words]
    return ' '.join(lemmatized)
```

```
In [8]: # Apply Lemmatization
lemmatized = lemmatize_text(cleaned_no_stop)
print("Lemmatized text:", lemmatized)
```

Lemmatized text: türkiyen başkent ankaradır istanbul kalabalık şehirdir doğal güzel
ik fazladır türk mutfağı dünyaca ünlüdür tarih kültür açısın zeng ülkedir

6. N-Gram Analizi

Unigram, bigram ve trigram oluşturma.

```
In [9]: def generate_ngrams(text, n):
        words = text.split()
        ngrams = []
        for i in range(len(words) - n + 1):
            ngram = ' '.join(words[i:i+n])
            ngrams.append(ngram)
        return ngrams
```

```
In [10]: # Generate unigrams (1-grams)
# Generate n-grams from Lemmatized text
unigrams = generate_ngrams(lemmatized, 1)
print("Unigrams:", unigrams[:10])

bigrams = generate_ngrams(lemmatized, 2)
print("\nBigrams:", bigrams[:10])
```

```
trigrams = generate_ngrams(lemmatized, 3)
print("\nTrigrams:", trigrams[:10])
```

Unigrams: ['türkiyen', 'başkent', 'ankaradır', 'istanbul', 'kalabalık', 'şehirdir', 'doğal', 'güzellik', 'fazladır', 'türk']

Bigrams: ['türkiyen başkent', 'başkent ankaradır', 'ankaradır istanbul', 'istanbul kalabalık', 'kalabalık şehirdir', 'şehirdir doğal', 'doğal güzellik', 'güzellik fazladır', 'fazladır türk', 'türk mutfağı']

Trigrams: ['türkiyen başkent ankaradır', 'başkent ankaradır istanbul', 'ankaradır istanbul kalabalık', 'istanbul kalabalık şehirdir', 'kalabalık şehirdir doğal', 'şehirdir doğal güzellik', 'doğal güzellik fazladır', 'güzellik fazladır türk', 'fazladır türk mutfağı', 'türk mutfağı dünyaca']

7. Sonuçlar ve Frekans Analizi

```
In [11]: # Count frequencies
unigram_freq = Counter(unigrams)
bigram_freq = Counter(bigrams)
trigram_freq = Counter(trigrams)

# Show most common
print("Most common unigrams:")
print(unigram_freq.most_common(5))

print("\nMost common bigrams:")
print(bigram_freq.most_common(5))

print("\nMost common trigrams:")
print(trigram_freq.most_common(5))
```

Most common unigrams:

```
[('türkiyen', 1), ('başkent', 1), ('ankaradır', 1), ('istanbul', 1), ('kalabalık', 1)]
```

Most common bigrams:

```
[('türkiyen başkent', 1), ('başkent ankaradır', 1), ('ankaradır istanbul', 1), ('istanbul kalabalık', 1), ('kalabalık şehirdir', 1)]
```

Most common trigrams:

```
[('türkiyen başkent ankaradır', 1), ('başkent ankaradır istanbul', 1), ('ankaradır istanbul kalabalık', 1), ('istanbul kalabalık şehirdir', 1), ('kalabalık şehirdir doğal', 1)]
```

```
In [12]: # Summary Report
print("="*50)
print("TEXT CLEANING & N-GRAM ANALYSIS REPORT")
print("="*50)
print(f"\nOriginal text length: {len(text)} characters")
print(f"Cleaned text length: {len(cleaned_advanced)} characters")
print(f"\nTotal unigrams: {len(unigrams)}")
print(f"Total bigrams: {len(bigrams)}")
print(f"Total trigrams: {len(trigrams)}")
print(f"\nUnique unigrams: {len(unigram_freq)}")
```

```
print(f"Unique bigrams: {len(bigram_freq)}")  
print(f"Unique trigrams: {len(trigram_freq)}")
```

```
=====
```

TEXT CLEANING & N-GRAM ANALYSIS REPORT

```
=====
```

Original text length: 172 characters
Cleaned text length: 166 characters

Total unigrams: 18
Total bigrams: 17
Total trigrams: 16

Unique unigrams: 18
Unique bigrams: 17
Unique trigrams: 16

In []: